Section 34
Run-Time Configuration

This section presents information about the configuration of hardware and software that run the ETMS. Section 34.1 describes the configuration of hardware used to implement the ETMS, with particular emphasis on the configuration at the Volpe Center. Section 34.2 discusses the processes that are invoked to run the ETMS.

34.1 ETMS Hardware Configuration

This section describes how equipment is configured for the ETMS at the Volpe Center. Figures 34-1 and 34-2 provide simplified diagrams of the operational ETMS system. The text that accompanies the diagrams highlights the main features of the ETMS hardware configuration. See Section 4.1 for more detailed descriptions of the ETMS hardware components.

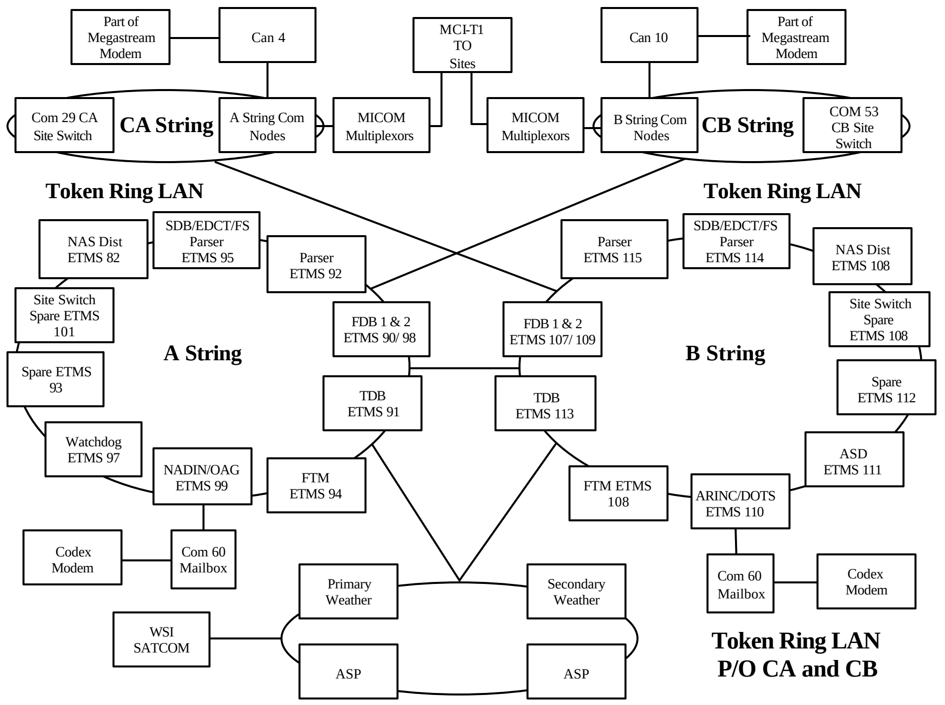

Figure 34-1. ETMS Hardware Configuration at Volpe Center

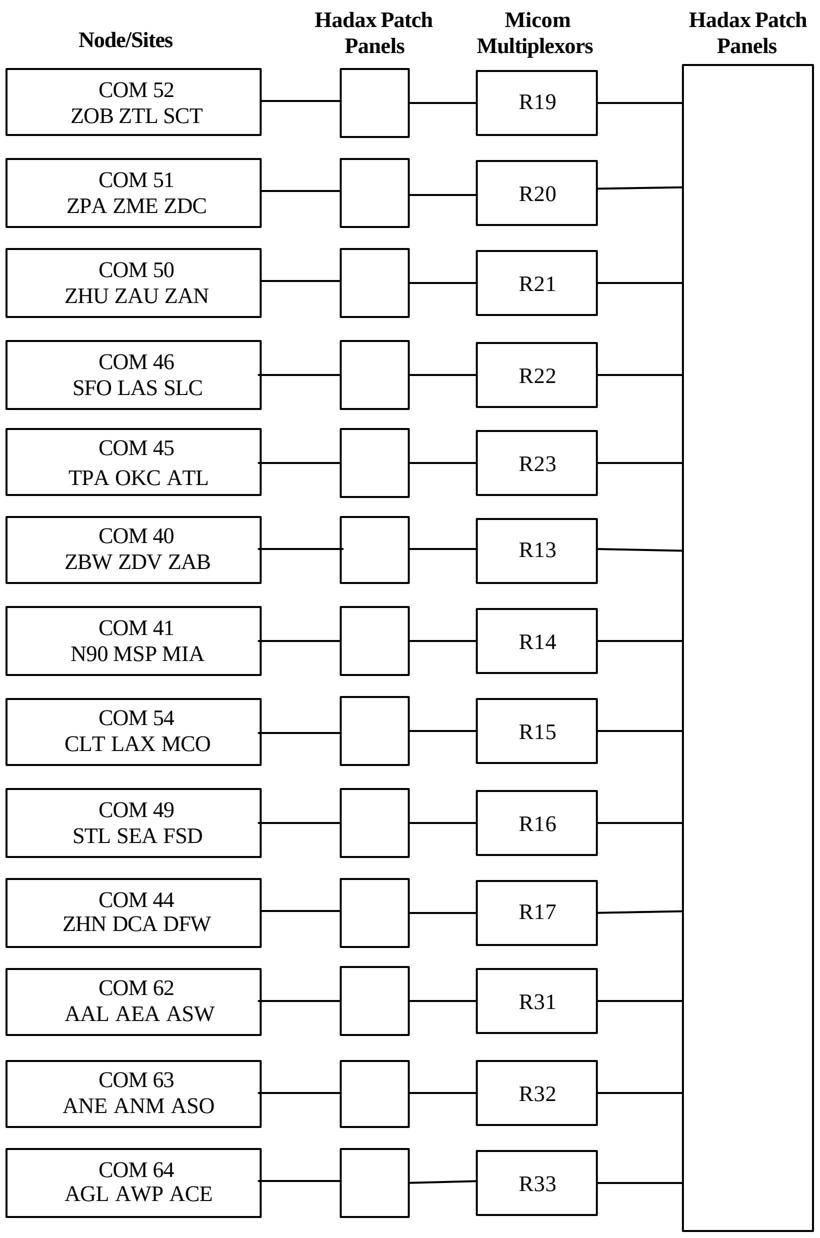

Figure 34-2. Hardware Configuration Details

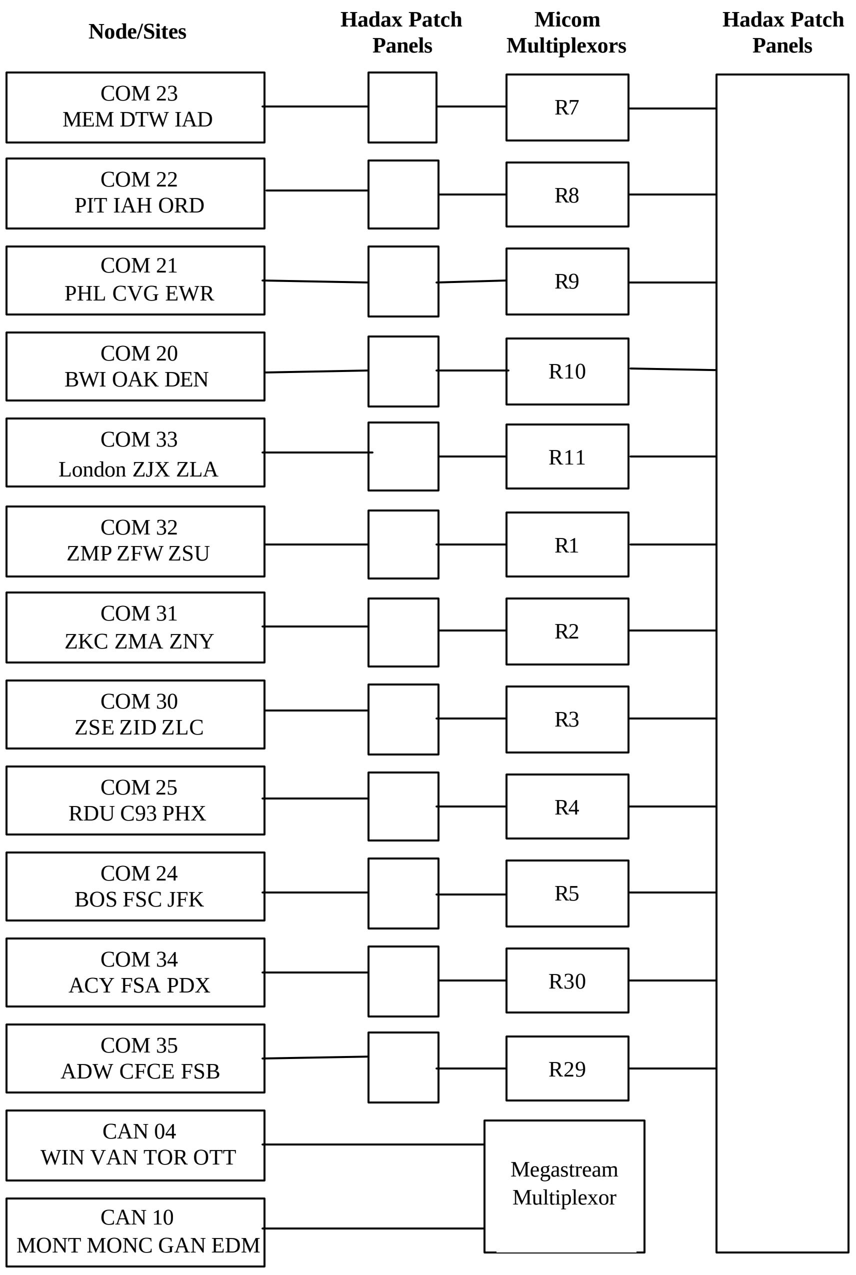

Figure 34-2. Hardware Configuration Details (continued)

34.1.1 Workstations

The following workstations are utilized for the ETMS system:

- Apollo 433s workstation with a high resolution color monitor
- Apollo 425t workstation with a high resolution color monitor
- Apollo DN4500/5500 workstations, each with a 19-inch black and white monitor

34.1.2 Operational Processing Hardware Configuration

The ETMS operational processing hardware consists of two sets, or *strings,* of Apollo workstations that are cabled together on separate *token ring* LAN connections to form the ETMS configuration. The strings are called *String A* and *String B*. Both strings are fully operational and are identical in configuration and data base storage. One string is always in the *master* mode, while the other string is in the *slave* mode. Either string can be in the *master* mode, and if there is a problem identified with the *master*, the *slave* string will assume the *master* function and will assign the other string to *slave* mode. Figure 34-1 illustrates the processor strings in detail. Also illustrated are the weather and Alert Server Process (ASP) feeds to the processing strings, as well as the processing strings feed to the communications strings and out to the remote sites. A more detailed description of each processor workstation, or *node,* on A and B string follows.

34.1.3 Operational Communications Strings and MICOM Multiplexors

Figure 34-1 illustrates a simplified drawing of the data flow from the processing strings to the A and B communication (COM) strings and Micom Multiplexors, and the feed to the MCI fiber optic T1 communication lines to the remote sites. The COM strings are named CA and CB strings, and are also on the token ring LAN. Unlike the processor strings, both COM strings are always in use. Each COM string has several spare nodes available in case of a node failure. Figure 34-2 illustrates a more detailed description of the connections between the COM nodes and the Micom Multiplexors. The left side of the drawing shows the COM node number and the remote sites that are assigned to that node. This figure depicts the normal site to COM node assignments; however, any site can be assigned to any COM node by software configuration. The Hadax Patch panel allows for manually monitoring the signal data flow between the node and the multiplexor. The panel also allows for manual patching of channels from one multiplexor to another in case of a hardware failure. The normal connection between the node and the multiplexor is straight through, with the sites listed with the COM node the same sites that pass through the numbered multiplexor to the MCI fiber optic lines.

34.1.4 Network Redundancy

The token ring LAN provides for several alternate paths of data flow from string to string. These connections, or Routers, utilize Apollo DN4500 and DN5500 workstations that physically link the strings. Each *Router* straddles two LANs, connecting the two networks and allowing the processes to communicate as if they were on the same network. There are six routers. Each Router supports communications between its primary strings, and also provides for alternate paths in the case of a failure.

(1) Between CA String and A String

(2) Between CB String and B String

(3) Between CA String and B String

(4) Between CB String and A String

(5) Between CA String and CB String

(6) Between A String and B String

34.1.5 Operational A and B String Node Descriptions

There are several minor differences in A and B String configurations. A String hosts the Watchdog Server on ETMS 97, while B String hosts the ASD Server on ETMS 111. A String hosts the NADIN and OAG processes on ETMS 99, which is the spare node for the ARINC and DOTS processes that run on B String ETMS 110. ETMS 110 is also the spare node for the NADIN and OAG. Figure 34-1 depicts their standard ETMS node assignment; however, each processor function can be assigned to other nodes within the string. A description of the operational A and B Strings and their associated nodes follows.

A String is configured with processes normally assigned as follows:

- Site Switch - COM 29.
- NAS Distribution - ETMS 82
- SDB/EDCT/FS Parser - ETMS 95
- Parser - ETMS 92

- FDB1 and FDB2 - ETMS 90/98
- TDB - ETMS 91
- FTM - ETMS 94
- NADIN and OAG - ETMS 99
- Watchdog - ETMS 97

B String is configured with processes normally assigned as follows (With the exceptions described in section 34.1.5, B String is configured the same as A String.) :

- Site Switch - COM 53
- NAS Distribution - ETMS 102
- SDB/EDCT/FS Parser - ETMS 114
- Parser - ETMS 115
- FDB1 and FDB2 - ETMS 107/109
- TDB - ETMS 113
- FTM - ETMS 108
- ARINC and DOTS - ETMS 110
- ASD - ETMS 111

A String. The A String processes are described in sections 34.1.5.1 to 34.1.5.9.

34.1.5.1 Site Switch — COM 29

The *Site Switch* is the main message switch of the *ETMS Network Addressing* function and is responsible for transferring messages among the nodes. Any message from a *Node Switch* is evaluated for destination address. If the address is on a node other than the originator, the *Site Switch* transfers the message to the appropriate *Node Switch.* The *Site Switch* contains a table with all the routing paths to all other ETMS *Site Switches*. When the *Site Switch* transfers a message to a node at another site, it uses the table for the correct routing path, and connects to that address through *Gate Switches* at each location and then to the correct *Node Switch.*

In summary, a process connects to its Node Switch, which connects to a Site switch. A message from one site to another goes from one Site Switch to a Node Switch, to the Gate Switch, across a communications medium to a remote Gate Switch, and to the Node Switch. Some remote sites may have a Site Switch to control the addressing at that location.

34.1.5.2 NAS Distribution — ETMS 82

The function of the *NAS Distribution* (*NAS Dist.*) is to receive data inputs from the following providers:

- NAS US flight position and flight plan messages
- NAS Canadian flight position and flight plan messages
- NAS London DZ, FZ, and TZ data
- DOTS TO data
- SDB Flight scheduling messages
- RT route messages

NAS Dist. formats the data and ships the appropriate data to registered clients. Within the Hub network, the NAS Dist. forwards data to the Q_Driver for input to the Parser node.

34.1.5.3 SDB/EDCT/FS Parser — ETMS 95

ETMS 95 node hosts three processes:

(1) *SDB* Process - This process provides airline schedule data to the ETMS system using the airlines OAG as a source of flight plan information. The *SDB Server* processes this OAG data and also accepts additional flight plan data from remote sites to update the *Update/Request Server* and the *SDB List Server*. The *Update/Request Server* creates FS messages (similar to FZ messages) from the flight schedules in the database. These messages are forwarded to the *NAS Dist*., which passes the data to the Traffic Model. The *SDB List Server* receives all LIST and CLIST commands from the *SDB Server* for all list requests made by the user through the *ASD* interface. When the list is ready, the *SDB Server* forwards it to the requesting listserver.

(2) *EDCT* Process - This process includes an *EDCT Manager* and an *EDCT Request Server* function. The *EDCT Manager* consists of a group of data-driven processes that receive control messages from user sites. The data is parsed and added to the Controls Database, and control information is sent to the Traffic Model function. The *EDCT Manager* also receives flight information from the Traffic Model to use

for developing and issuing Control Time messages. The *EDCT Request Server* services user requests regarding control related live data. It receives user requests and after processing through the Controls Database, sends the output to the requesting user and, in some conditions, to the *Apply Ground Delays* function. The *EDCT Server* processes TM Commands, which may alter the existing database, and TM Data Requests, which look for information from the existing database.

(3) *FS Parser* - This function performs parsing of the Schedule Flight Plans (FS) and scheduled flight plan cancellations (RS). It is part of the logical *Parser* process that resides on ETMS 92.

34.1.5.4 Parser — ETMS 92

The *Parser* function is designed to extract, convert, and pass input data and forward it to the Flight Database. The most notable function is the conversion of flight path information into an event list. The *Parser* on ETMS 92 processes the following subsets of *NAS Dist.* packets.

- NAS messages
- Test messages
- Flight path data from the Flight Database Processor (FA)

The FS Parser hosted on ETMS 95 processes flight plan schedules and cancellations. The Parser consists of four processes.

(1) *Queue_Driver* - This process reads data packets sent from the *NAS Dist*. and enqueues valid packets for the *Parser* process. The input message is an array filled with many separate messages, which are separated from one another by a linefeed. The valid data packets are forwarded to the *Parser* and *FS Parser*, as appropriate.

(2) *Parser* - The function of the *Parser* is to break down each of the message packets into its constituent parts, convert data to the correct internal format, and pass the parsed data to the *Flight Database Processor*. The *Parser* reduces the queued data stream into single messages, converts them as necessary, and forwards them to the *FDB* after the entire message has been parsed.

(3) *Parser.relay* - This process is used to provide queueing operations between different nodes across the network to the *FDB processor*.

(4) *Feedback.receiver* - This process provides communications from the *FDB* to the *Parser* process. Data previously stored in the flight database is passed back to the *Parser* by this process and is used to aid in flight plan route conversions.

34.1.5.5 FDB1 and FDB2 — ETMS 90/98

The *Flight Database Process (FDB)* performs the following functions:

- Maintains an entry for each active, proposed, or completed (within the last 12 hours) flight in the NAS system
- Updates flight records when new data arrives
- Distributes all updates to other functions within the ETMS system

Due to the large amount of data that must be stored, two processing nodes are used to store the flight information. *FDB1* processes data for flight IDs that begin with A-M, and *FDB2* processes data for flight IDs that begin with N-Z. Since there is a *master/slave* relationship between the FDBs of the A and B strings, both sets of FDBs are linked together to maintain unique indices across the strings. The *master* string is responsible for processing all incoming flight messages. Once the message has been processed and stored in the FDB, the *master* sends a copy of the stored data to the *slave* string FDB. The *slave* string receives only updates, in the form of FDB messages from the *master* string, and performs a database update on that input. This process ensures that the database entries for any particular flight are identical in both *master* and *slave* strings. In the case of one string failing and losing all of the data in the FDB, the second string can perform a Cross String Recovery, which copies all of the present FDB data from the good string into the second string FDB.

34.1.5.6 TDB — ETMS 91

The *TDB* monitors air traffic demands in real time for a large number of NAS elements such as airports, fixes, and sectors. The *TDB* generates air traffic reports automatically and on-request to support the Monitor/Alert features in the *ASD.* The *TDB* is divided into three subsystems: airports, fixes, and sectors. Each database keeps two tables of information for a time period of from 24 hours in the past to 40 hours in the future. The two tables are the Statistics and Flights tables.

(1) Statistics table - contains traffic demands, capacities, and General Aviation traffic estimates

(2) Flights table - contains pointers to flight lists by interval and element

The *TDB* checks every date-time pair in the input for compliance with the current time window except for inputs from the *TDB* transactions data stream. The *TDB* performs flight updates in the database such as add flight, delete flight, and replace flight. Capacity updates, GA estimates, and alert color updates are also part of the Update Traffic Data.

34.1.5.7 FTM — ETMS 94

The *Flight Table Manager (FTM)* function is to maintain a database containing flight information on all flights operating under FAA Instrument Flight Rules (IFR) and to provide aircraft position updates to the *ASD*. The *FTM* also provides data to on-request programs. The *FTM* database information includes flight plans, flight plan changes, position reports, arrival data, departure data, and route data.

The *FTM* function is composed of the *FTM* and *FTM_Coprocess* processes. The *FTM process* requests and receives data from *FDBD*, processes the data, updates its database, produces map files, and returns replies. The *FTM_Coprocess* receives requests for reports from the *FTM* and after processing, forwards the reply to the requesting process. Both processes have read access, but only the *FTM* has **write** access. If the *FTM_Coprocess* cannot function, the *FTM* produces the desired reports.

The *FTM* also performs ETMS Network Address interfacing, statistics collection, user request processing, reconfigure processing, recovery data processing, raw data parsing, backup *FTM* implementation, error logging, and disk archiving.

34.1.5.8 NADIN and OAG — ETMS 99

ETMS disseminates flow control messages to national and international flow control facilities through the National Airspace Data Interchange Network (NADIN). ETMS also receives various messages of note through this interface. The *NADIN/ARINC Server* is a dual program that functions for either process. The code is virtually identical, with the exception of address length, priority field format, message length, and message framing. The Server receives block messages from the driver and passes the data to the appropriate clients. It also forwards messages from clients or mailboxes to the NADIN/ARINC network. Each message has its destination address checked, and if it is valid, the message is sent to the user's address. The Server maintains an auto distribute list in its configuration file. The destination NADIN/ARINC address is checked, and if valid, the message is sent.

Each message being output is queued into a permanent file that is reused. The message is removed from the queue under three circumstances:

(1) It was successfully transmitted.

(2) It timed out.

(3) The queue overflowed.

The last two cause the message to be returned to the sender, an error message sent to the logger, and an entry made into the current hour's XMT file.

The *NADIN/ARINC Printer* Function moves received messages from the server to the printers at remote sites. The function uses three modules to accomplish this move.

(1) Sprinter module - the Spooler

(2) Rprinter module - the remote queue manager

(3) Qprinter module - the queue printer

This description applies as well to ARINC (See section 34.1.5.10) on B String.

OAG ETMS receives weekly updates of scheduled flight plans from the Official Airline Guide (OAG) interface. This data is used to update the SDB. This is a receive-only interface.

34.1.5.9 Watchdog — ETMS 97

The primary *Watchdog (WD)* function is to monitor the communications lines between the hubsite and all field sites. The program issues probe commands to the sites and measures the time it takes for the sites to respond. The en route time determines what color is displayed for that site for any specific probe. The green color represents a quick return. Yellow is a slight slowing. Orange is a very slow return. Red is an unreturned probe. The times are user defined and are variable. The process also monitors the NAS data flow,and displays a blinking purple followed by a normal color for any NAS site not providing data. A brown color indicates that the *WD* is skipping the site.

The site window also displays whether the primary or secondary hubsite String assigned for each remote site is on line. An upper case letter after the site name indicates the primary string on line, and a lower case letter indicates the secondary string.

There is also a phone book database that has the contacts list for each site.

The *WD* has a server and client process. The server runs at the hubsite, and provides the data to all other locations.

B String. The B String processes are described in sections 34.1.5.10 to 34.1.5.11. With the exceptions described in section 34.1.5, B String is configured the same as A String.

34.1.5.10 ARINC and DOTS — ETMS 110

ETMS disseminates flow control messages to the airlines through the Aeronautical Radio Incorporated (ARINC) network. The airlines can also send substitute or insert messages to the ETMS. The ETMS responds with SI messages as appropriate. See section 34.1.5.8 for additional information on the process.

DOTS. DOTS provides oceanic position reports for ARINC.

34.1.5.11 ASD — ETMS 111

The ASD provides users with a graphical display of current aircraft positions for all active IFR flights on a national scale. It can superimpose geographical boundaries, NAS facilities, special use areas, and NAVAIDS. The ASD can display weather, Monitor Alerts, Reroute Information Reports, EDCT information, and various airline and flight databases. The ASD is hosted on ETMS 111 at the hubsite and can be called up from any node at the site. Each remote site hosts an ASD program which can be tailored to the specific needs of each site. All of database nodes described in this section provide data for the ASD.

34.1.6 Weather Strings

The primary weather string (WSI 01, 02, 03) provides weather to the operational systems, and the secondary weather string (WSI 11, 12, 13) provides weather data to the in-house development and test systems. Either string can be assigned to either function, as both strings are identical configurations, and provide the same data outputs.

WSI provides a satellite link and dish as the primary input to the system. A SATCOM receiver splits the feed for both strings, and feeds a PC Data Acquisition System (DAS), which provides the interface to format the data for the Apollo Ingest node, WSI 01 and 11. The Ingest node parses the data into the NEXET, NOWRADHF, LIGHTNING, TAF, METAR, STORMCAST+, and RUC211 components. and sends the data to the Product Generation node, WSI 03 and 13. This node creates the graphics, and it breaks the data down into its sub-components. This node hosts the MOVDIR, which sends the files to the locations in its address table. The following weather components are sent to the addressed users: Jet_Stream, Winds_loft, Metar, TAF, Lightning, NOWRAD6, NOWRAD_2km, NOWRAD_LAB, echo_tops, NWS_STATS, NWS_CHANGE, and NWS_ADMIN. The File Server for the strings are hosted on WSI 02 and 04. The RUC211 data is sent to the CTAS only.

The WSI 01, 02, 03 nodes share the CA TOKEN RING LAN, and WSI 11, 12, 13 nodes share the CB TOKEN RING LAN. A backup dial-up modem connection to each DAS PC automatically goes on line when the Satellite link fails to provide weather data. All but winds data is available when the system is using the modem.

34.1.7 Alert Server Processor

The *Alert Server Processor (ASP*) is hosted on both weather strings so that the process will be active whichever string is assigned as primary. WSI 03 and 13 host the *ASP*. The *ASP* maintains the status and demand counts of all alerted elements and provides this data to the *ASD* at all sites for use in displaying traffic demands for fixes, sectors, or airports.

34.2 ETMS Software Configuration

This section describes the software that runs the ETMS functions on the ETMS A and ETMS B nodes at the Volpe Center. In general, a node on ETMS A is configured with the same processes as its functional counterpart on ETMS B. The software configuration described in this section is organized by ETMS processes. Complete pathnames from the node root (/) directory are given for each file. The following list generally represents configurations for both strings.

- Site Switch/NAS/FTM/Weather/ASP
- EDCT/SDB
- Store Flushed Flights/Delay Advisor
- Parser
- FDB
- TDB
- Communications

34.2.1 Site Switch

The *site.sw* runs the major ETMS *Network Addressing* processes.

Processes

The following processes run on the *site.sw*:

site.sw — This is the central message switching process for transferring messages among the nodes on a site.

node.sw — This process is responsible for transferring messages between processes on a node. Most ETMS processes on a node connect to the node.sw to send messages to other ETMS processes.

Nodescan — This process monitors the execution of specified ETMS processes on a node. If a process is not found, it will be restarted by Nodescan.

Purger — This process regularly deletes files which are older than a specified time interval.

wx.server — This process receives METAR and TAF weather data files, processes the file into reports and logs the data to disk files. It also receives weather data requests from local/remote sites, processes the requests, and returns the information to the requesting process.

security_monitor — This process periodically polls all FTP and CMD processes on the site to ensure that all are using the correct password file. When a security violation is detected, it broadcasts an alarm message to all Net.mail processes.

Rmgr — the route manager server process is started by Nodescan

rawlog — a process that allows for the insertion or correction of flight data in the ETMS databases

logger — a process used to probe remote sites. It also records system messages sent to it.

wdg_s — a process that monitors remote site communications. Works in conjunction with a watchdog client.

wb_server — a process which allows for simultaneous viewing of a common graphic file between multiple users.

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

File: /etms/nodescan/config/nodescan.config

```
!
@
%
&
```

```
|
?
#               end               of               macro               definitions
#
#
*   ---  node.sw                 /etms/node.sw/node.sw       /etms/node.sw/config/node.sw.config
#
*   ---  ftp1                    /etms/ftp/ftp
#
*          ---        ftp2                                                    /etms/ftp/ftp
#
*   ---  cmd                     /etms/cmd/cmd
#
*   ---  purger                  /etms/purger/purger          /etms/purger/config/purger.config
#
*   ---  lsthub                  /etms/list/lstnet
#
#*        ---    gate.sw.c93       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.c93.svr
#
#*        ---    gate.sw.den       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.den.svr
#
#*      ---   gate.sw.london     /etms/gate.sw/gate.sw   /etms/gate.sw/config/gate.sw.config.london.svr
#
*         ---    gate.sw.okc       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.okc.svr
#
*         ---    gate.sw.zfw       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zfw.svr
#
*         ---    gate.sw.zid       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zid.svr
#
#*        ---    gate.sw.zjx       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zjx.svr
#
*         ---    gate.sw.zkc       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zkc.svr
#
*         ---    gate.sw.zla       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zla.svr
#
#*        ---    gate.sw.zlc       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zlc.svr
#
*         ---    gate.sw.zma       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zma.svr
#
*         ---    gate.sw.zmp       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zmp.svr
#
*         ---    gate.sw.zny       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zny.svr
#
*         ---    gate.sw.zse       /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zse.svr
#
*   --- gate.sw.zsu   /etms/gate.sw/gate.sw /etms/gate.sw/config/gate.sw.config.zsu.svr
```

34.2.2 NAS Node

The NAS processes are responsible for receiving NAS and other ETMS data and distributing the data to various ETMS processes. This includes the control time messages sent to the HOST interface.

Processes

The following processes run on the NAS:

NASC - This process receives NAS messages from ETMS remote sites, validates the format, collates, and filters the data (if requested). The data is then forwarded to specified destinations.

nas.dist - This process receives NAS and other data from providers and formats, validates the time stamps, and forwards to registered client processes.

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

File: /etms/nodescan/config/nodescan.config

```
!
@
%
&
|
?
#          end          of          macro          definitions
#
#
*  ---  node.sw              /etms/node.sw/node.sw       /etms/node.sw/config/node.sw.config
#
*  ---  ftp1                 /etms/ftp/ftp
#
*      ---     ftp2                                                       /etms/ftp/ftp
#
*  ---  cmd                  /etms/cmd/cmd
#
*  ---  purger               /etms/purger/purger         /etms/purger/config/purger.config
#
*  ---  lsthub               /etms/list/lstnet
#
#*     ---   gate.sw.c93     /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.c93.svr
#
#*     ---   gate.sw.den     /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.den.svr
#
```

```
#*        ---   gate.sw.london        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.london.svr
#
*         ---   gate.sw.okc           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.okc.svr
#
*         ---   gate.sw.zfw           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zfw.svr
#
*         ---   gate.sw.zid           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zid.svr
#
#*        ---   gate.sw.zjx           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zjx.svr
#
*         ---   gate.sw.zkc           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zkc.svr
#
*         ---   gate.sw.zla           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zla.svr
#
#*        ---   gate.sw.zlc           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zlc.svr
#
*         ---   gate.sw.zma           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zma.svr
#
*         ---   gate.sw.zmp           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zmp.svr
#
*         ---   gate.sw.zny           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zny.svr
#
*         ---   gate.sw.zse           /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zse.svr
#
*   --- gate.sw.zsu   /etms/gate.sw/gate.sw /etms/gate.sw/config/gate.sw.config.zsu.svr
```

34.2.3 SFF/DEL_ADV

This section lists the processes and nodescan configuration file contents for the *STORE FLUSHED FLIGHTS* (SFF) and *DELAY ADVISOR*. The SFF holds all flights and the associated data which is flushed from the DFB for later use in building the Ground Time Prediction database for the *DELAY ADVISOR* and *FLIGHT DATABASE*.

Processes

The following processes run on the *SFF/DEL_ADV*:

store_flushed_flts -— stores flushed flight data in history files.

sff_receiver — receives flushed flight data from the FDB via the das.relay.

del_adv — the delay advisor process provides ground time reports upon request. This named process is started by Nodescan.

(delay_advisor) — the UID process for the delay advisor

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

Filename: /etms/startup/parameters/node_scan_list.sff

```
*     ---   del_adv          /etms/api/starter          /etms/startup/parameters/del_adv.ss.input
*  --- sff      /etms/api/starter    /etms/startup/parameters/store_flushed_flts.ss.input
```

34.2.4 EDCT/SDB

This section lists the processes and *Nodescan* file contents for the *EDCT/SDB*. The processes are the *EDCT, SDB, FS Parser,* and *Delay Advisor*. The *EDCT* process maintains a database of flight controls. The *SDB* generates schedule messages, which the *Parser* interprets. The *Delay Advisor* provides ground delay reports upon request.

Processes

The following processes run on the *EDCT/SDB*:

edct_driver — the main EDCT process responsible for processing delay programs and launching its subordinate processes. This named process is started by Nodescan.

(edctdriver) — the UID process for the EDCT driver

(edctfdbrelay) — the edctfdb relay process forwards delay information to the FDB. This process is started by the EDCT driver.

(fdbedct.receiver) — the FDBEDCT receiver process accepts flight updates for controlled flights from the FDB. This process is started by the EDCT driver.

(tdbedct_tap.receiver) — the TDBEDCT tapping receiver process accepts flight lists from the TDB server. This process is started by the EDCT driver.

edct_req_svr — the EDCT request server process provides reports on demand. This named process is started by Nodescan.

(edct_req_server) — the UID process for the EDCT request server

archive — the archive process saves selected EDCT files to assist in troubleshooting. This process is started by Nodescan.

ifcn.fe — the IFCN front end process is started by Nodescan

RTR — the ETMS Router process is started by Nodescan

(Nodescan) — This process monitors the execution of specified ETMS processes on a node. If a process is not found, it will be restarted by Nodescan.

update_request_server — the SDB update request server process generates schedule messages and sends them to the NAS distributor.

sdb_server — the SDB server process coordinates intra-SDB communication

list_server — the SDB request server process produces SDB reports upon request from the SDB server process

nodescan.sdb—- This process monitors the execution of specified ETMS processes on a node. If a process is not found, it will be restarted by Nodescan.

Ctfwrd — This process receives CT messages from EDCT via ifcn.fe and forwards the CTs to the nas.server at the specified ETMS remote site.

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

/etms/startup/parameters/node_scan_list.edct

```
*   ---  edct_driver     /etms/api/starter   /etms/startup/parameters/edctdriver.ss.input
*   ---  edct_req_svr    /etms/api/starter   /etms/startup/parameters/edct_req_server.ss.input
*  --- ifcn.fe       /etms/ifcn.fe/ifcn.fe /etms/ifcn.fe/config/ifcn.fe.config /etms/ifcn.fe/data/address_table
/etms/ifcn.fe/data/center_table
*    ---   rtr                                                              /etms/rtr/rtr
*  --- ctfwrd       /etms/ctfwrd/ctfwrd /etms/ctfwrd/config/ctfwrd.config
```

34.2.5 Parser

This section lists the processes and *Nodescan* startup file contents for the *Parser*. The *Parser* is responsible for interpreting the raw NAS and FS data and forwarding the processed data to the *FDB.*

Processes

The following processes run on the *Parser*:

queue_driver — the queue driver process which receives NAS and FS data from the NAS distributor via network addressing.

(queue_driver) — the UID process for the queue driver.

parser — the parser process which interprets the NAS and FS data and supplies it to the FDB via the parser relay.

(parser) — the UID process for the parser.

(parser.relay) — the parser relay process is started by the parser and forwards processed nas data to the FDB.

(feedback.receiver) — the feedback receiver is started by the queue driver to receive feedback messages from the FDB.

route_db.relay — allows the route.db process to receive route information.

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

/etms/startup/parameters/node_scan_list.parser

```
*        ---    queue_driver              /etms/api/starter     /etms/startup/parameters/queue.ss.input
*  --- parser        /etms/api/starter /etms/startup/parameters/parser.ss.input
```

34.2.6 FDB

This section lists the processes and *Nodescan* startup file contents for the *FDB*. The *FDB* receives processed flight and schedule data from the *Parser*, updates its database with the data, and forwards the data to the *TDB* and *FTM*.

Processes

The following processes run on the *FDB* node:

fdb — the main FDB process responsible for launching communication processes and maintaining its database. This named process is started by Nodescan.

(fdb_manager) -—the UID process for the FDB

(decoupled_receiver) — the FDB receiver process started by the FDB manager.

(tdbrelay) — the TDB relay process forwards transactions to the TDB. This process is started by the FDB manager.

(ftmrelay) — the FTM relay process forwards transactions to the FDB distributor receiver process. This process is started by the FDB manager.

(feedbackrelay) — the feedback relay process sends transactions to the Parser. This process is started by the FDB manager.

(dasrelay) — the dasrelay process forwards transactions to the store flushed flight's receiver process. This process is started by the FDB manager.

(edctsvrrelay) — the EDCT server relay process forwards transactions to the EDCT process. This process is started by the FDB manager.

(rt_relay) — the route relay process forwards RT messages to the nas distributor using network addressing. This process is started by the FDB manager.

(cross_string_relay) — the cross string relay process which forwards FDB database updates to the slave string. This process is started by the FDB manager.

router — the FDB Router process. This named process is started by Nodescan

(fdb_router) — the UID process for the FDB Router

data_svr — the FDB Data Server process. This named process is started by Nodescan

(fdb_data_server) — the UID process for the FDB Data Server

(ftm_recovery) — the FTM Recovery process is an ephemeral process started by the FDB data server.

(fdb_distributor) — the FDB Distributor process sends processed flight data to the corrected FTM.

(fdb_recovery) — the FDB Recovery process is an ephemeral process started by the FDB data server. It provides for master to slave cross string recovery.

showq_etms98 — the show queue process reports the status of the FDB queues. This named process is started by Nodescan.

(show_queue) — the UID process for show queue

node.sw — This process is responsible for transferring messages between processes on a node. Most ETMS processes on a node connect to the node.sw to send messages to other ETMS processes.

time.check — This process periodically verifies the node time with the fileserver node time.

Purger — This process regularly deletes files which are older than a specified time interval.

nodescan.fdb — This process monitors the execution of specified ETMS processes on a node. If a process is not found, it will be restarted by Nodescan.

route.db — This process collects route information for use in the weekly SDB build.

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

/etms/startup/parameters/node_scan_list.fdb

```
*    ---  fdb_1          /etms/api/starter      /etms/startup/parameters/fdb_driver.ss.input
*    ---  router         /etms/api/starter      /etms/startup/parameters/fdb_router.ss.input
*    ---  data_svr_1     /etms/api/starter      /etms/startup/parameters/fdb_data_svr.ss.input
*    ---  fdbd           /etms/api/starter      /etms/startup/parameters/fdb_dist.ss.input
*  --- rdb    /etms/api/starter   /etms/startup/parameters/route_db.ss.input
```

34.2.7 TDB

This section lists the processes and *Nodescan* startup file contents for the *TDB* node. The *TDB* receives processed flight data from the *FDB*, updates its databases with the data, generates periodic Monitor/Alert reports, and provides request services on demand.

Processes

The following processes comprise the *TDB*:

tdb_mgr — the main TDB process responsible for launching subordinate database and communications processes. This process is started by Nodescan.

(decoupled_receiver) — the TDB receiver process started by the TDB manager

(uni_airports) — the TDB airport update process started by the TDB manager

(uni_fixes) — the TDB fixes update process started by the TDB manager

(uni_sectors) — the TDB sector update process started by the TDB manager

(tdb_server) — the TDB server process coordinates request service. This process is started by the TDB manager

airport_svr — the airport data server process provides reports upon request. This named process is started by Nodescan.

(airports_server) — the UID process for the airports data server

fix_svr — the fix data server process provides reports upon request. This named process is started by Nodescan.

(fixes_server) — the UID process for the fixes data server sector_svr — the sector data server process provides reports upon request. This named process is started by Nodescan.

(sectors_server) — the UID process for the sectors data server

tdb_eval — the TDB evaluation server process launches and coordinates the activities of the evaluation processes. This named process is started by Nodescan.

(evaluation_server) — the UID process for the evaluation server

(airports_evaluation) — the airports evaluation process started by the evaluation server

(fixes_evaluation) — the fixes evaluation process started by the evaluation server

(sectors_evaluation) — the sectors evaluation process started by the evaluation server

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

/etms/startup/parameters/note_scan_list.tdb

```
*   ---  tdb_mgr      /etms/tdb/programs/uni_distribution  /etms/tdb/parameters/parudis  -p  -d  /etms/tdb  -e
*    ---   tdb_eval        /etms/api/starter      /etms/tdb/parameters/start.eval_svr     -i
*    ---   fix_svr         /etms/api/starter      /etms/tdb/parameters/start.fixes_svr    -i
*    ---   sector_svr      /etms/api/starter      /etms/tdb/parameters/start.sectors_svr  -i
*    ---   airport_svr     /etms/api/starter      /etms/tdb/parameters/start.airports_svr -i
#*         ---   xpndr                                                  /etms/xpndr/xpndr
#*  --- rmgr      /etms/rmgr/rmgr
```

34.2.8 FTM

The *FTM* runs processes responsible for maintaining the flight database.

Processes

The following processes run on the *FTM*:

FTM — The Flight Table Manager process maintains a database containing flight information on all flights operating under FAA IFR and provides timely updates of aircraft positions to the ASD. It also provides flight data to various processes upon request.

ftm_coproc — This process handles flight data report requests for FTM.

ftm_stats — This process provides a dual window display containing status information about the FTM and the communications ports.

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

File: //etms94/etms5/nodescan/config/nodescan.config

```
*  ---  ftm            /etms/ftm/ftm              /etms/ftm/config/ftm.config
#
*  ---  ftm_stats      /etms/ftm/ftm_stats                                          etmsa
#
*  ---  ftm_coproc     /etms/ftm/ftm_coproc
#
```

34.2.9 Weather/ASP Node

The *Weather/ASP* runs the *Weather* and *Monitor Alert* processes.

Processes

The following processes run on the *Weather/ASP*:

Wxsend — This process transfers weather data files received from movedir to the ETMS remote sites.

ASP — The Alert Server Process provides traffic alert reports generated by the TDB to the TSD to be used in Monitor Alert.

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

File: /etms/nodescan/config/nodescan.config

```
!
@
%
&
|
?
#          end          of          macro          definitions
#
*  ---  purger              /etms/purger/purger          /etms/purger/config/purger.config
#
```

```
*  ---  node.sw          /etms/node.sw/node.sw              /etms/node.sw/config/node.sw.config
#
*  ---  site.sw          /etms/site.sw/site.sw              /etms/site.sw/config/site.sw.config
#
*  ---  ftp1             /etms/ftp/ftp
#
*  ---  ftp2             /etms/ftp/ftp
#
*  ---  cmd              /etms/cmd/cmd
#
*  ---  sec_monitor      /etms/ftp/sec_monitor
#
*  ---  rawlog                     /etms/rawlog/rawlog
#
*  ---  nas.dist         /etms/nas.dist/nas.dist            /etms/nas.dist/config/nas.dist.config
#
*  ---  nasc             /etms/nas/nasc                              /etms/nas/config/nasc.config
#
*  ---  wx.server        /etms/wx.server/wx.server  /etms/wx.server/config/wx.config
/etms/wx.server/data/locid_file
#
*  ---  wxsend                     /etms/wxsend/wxsend         /etms/wxsend/config/wxsend.config
#
*  ---  asp              /etms/asp/asp
          /etms/asp/config/asp.config
#
*  ---  rmgr             /etms/rmgr/rmgr
#
*  ---  dots             /etms/dots/dots                             /etms/dots/config/dots.config
#
*  ---  lstnet           /etms/list/lstnet
#
*  ---  logger                     /etms/logger/logger
#
*  ---  ftm              /etms/ftm/ftm
/etms/ftm/config/ftm.config
#
*  ---  ftm_stats        /etms/ftm/ftm_stats                                               etmsa
#
*  ---  ftm_coproc       /etms/ftm/ftm_coproc
#
*  ---  wdg_s            /etms/wdg/wdg_s
#
*  ---  wb_server        /etms/wb/wbserv
```

34.2.10 Communications Nodes

The *Communications* nodes run all the software needed for communications to remote sites. Typically, 16 remote sites are supported by one hubsite *Communications* node.

Processes

The following processes run on a *Communications* node:

node.sw — This process is responsible for transferring messages between processes on a node. Most ETMS processes on a node connect to the node.sw to send messages to other ETMS processes.

Nodescan — This process monitors the execution of specified ETMS processes on a node. If a process is not found, it will be restarted by Nodescan.

Purger — This process regularly deletes files which are older than a specified time interval.

gate.sw — This process performs the interface to the communications software, Xfer. It is responsible for routing data from one site to another. For sites that are reachable via communications lines, gate.sw passes messages to and from the Xfer processes. If the sites reside on the same LAN, gate.sw passes the messages via another gate.sw.

FTP — The file transfer process is used to transfer files between ETMS sites.

Cmd — This process is used to perform remote shell commands at ETMS sites.

Lstnet — This process provides ETMS users with specified flight information in report format.

NODESCAN File

The following is the *Nodescan* configuration filename and contents:

File: /etms/nodescan/config/nodescan.config

```
!
@
%
&
|
?
#                    end                    of                    macro                    definitions
#
#
*   ---  node.sw                        /etms/node.sw/node.sw          /etms/node.sw/config/node.sw.config
```

```
#
*   ---  ftp1                          /etms/ftp/ftp
#
*          ---       ftp2                                                  /etms/ftp/ftp
#
*   ---  cmd                           /etms/cmd/cmd
#
*   ---  purger                              /etms/purger/purger     /etms/purger/config/purger.config
#
*   ---  lsthub                        /etms/list/lstnet
#
#*         ---    gate.sw.c93        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.c93.svr
#
#*         ---    gate.sw.den        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.den.svr
#
#*       ---   gate.sw.london     /etms/gate.sw/gate.sw   /etms/gate.sw/config/gate.sw.config.london.svr
#
*          ---    gate.sw.okc        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.okc.svr
#
*          ---    gate.sw.zfw        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zfw.svr
#
*          ---    gate.sw.zid        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zid.svr
#
#*         ---    gate.sw.zjx        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zjx.svr
#
*          ---    gate.sw.zkc        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zkc.svr
#
*          ---    gate.sw.zla        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zla.svr
#
#*         ---    gate.sw.zlc        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zlc.svr
#
*          ---    gate.sw.zma        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zma.svr
#
*          ---    gate.sw.zmp        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zmp.svr
#
*          ---    gate.sw.zny        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zny.svr
#
*          ---    gate.sw.zse        /etms/gate.sw/gate.sw    /etms/gate.sw/config/gate.sw.config.zse.svr
#
*   --- gate.sw.zsu   /etms/gate.sw/gate.sw /etms/gate.sw/config/gate.sw.config.zsu.svr
```